제 25장 클래스

문법적 설탕

클래스가 도입되었다고 해서 기존 객체지향 모델을 폐기한것은 아니구 클래스 기반 패턴처럼 사용할수 있게한 **문법적 설탕**이라 볼수 있다

고로 **새로운 객체 생성 매커니즘**으로 이해하면 좋다

생성자 함수 vs 클래스

1. new 연산자 사용
2. extends, super 사용
3. 호이스팅이 발생하지않는것처럼 사용
4. strict mode 강제 사용
5. 열거 불가 → Enumerable = false

클래스의 정의

```
class Person {}

const MinJUN = NEW Person {} ;
```

클래스의 특징

- 무명의 리터럴로 생성 가능
- 변수나 자료구조에 저장 가능
- 함수의 매개변수에게 전달 가능
- 함수의 반환값으로 사용

클래스에 정의 할수 있는 메서드는

1. construtor(생성자)
2. 프로토타입 메서드

3. 정적 메서드

총 3가지 이다.

construtor 메서드

```
// 클래스
class Person {
  // 생성자
  constructor(name) {
    // 인스턴스 생성 및 초기화
    this.name = name;
  }
}

// 생성자 함수
function Person(name) {
  // 인스턴스 생성 및 초기화
  this.name = name;
}
```

인스턴스를 생성하고 초기화 하기 위한 특수한 메서드이다.

내부의 this는 클래스가 생성한 인스턴스를 가리킨다.

- 인스턴스의 construtor메서드

  인스턴스의 어디에도 construtor메서드가 보이지않는다

이는 construtor는 메서드로 해석되지않고 클래스가 평가되어 생성한 함수객체코드의 일부가 된다는걸 설명한다.

- construtor의 특징
  - construtor는 생략가능하다
  - construtor는 클래스내에 최대 한개만 존재가능하다,
  - 인스턴스를 생성할때 초기값을 전달할수 있다 .
    - 전달시에 매개변수로 전달한다
  - 별도의 반환문을 가지면 안된다

프로토타입 메서드

```
class Person {
  // 생성자
  constructor(name) {
    // 인스턴스 생성 및 초기화
    this.name = name;
```

```
  }

  // 프로토타입 메서드
  sayHi() {
    console.log(`Hi! My name is ${this.name}`);
  }
}

const me = new Person('Lee');
me.sayHi(); // Hi! My name is Lee
```

기존의 생성자함수와 다르게 prototype 프로퍼티에 메서드를 추가하지않아도

기본적으로 프로토타입 메서드로 된다.

정적 메서드

```
class Person {
  // 생성자
  constructor(name) {
    // 인스턴스 생성 및 초기화
    this.name = name;
  }

  // 정적 메서드
  static sayHi() {
    console.log('Hi!');
  }
}
```

인스턴스를 생성하지 않아도 호출할수 있는 메서드

static 키워드를 붙여서 선언한다.

정적메서드와 프로토타입 메서드의차이

1. 자신이 속해있는 프로토타입 체인이 다르다
2. 정적클래스는 클래스로 호출 , 포로토타입메서드는 인스턴스로 호출

3. 정적클래스는 인스턴스 프로퍼티 참조불가

클래스 필드 정의

클래스가 생성할 인스턴스이 프로퍼티

this를 생략해도 클래스필드를 참조가능하다

상속에 의한 클래스 확장

기존 클래스를 상속받아 새로운 클래스로 확장하여 정의하는것

1. 생성자 함수도 상속가능
    - 클래스 뿐 아니라 생성자함수도 상속가능하다
2. extends 키워드
    - 상속을 통해 클래스를 확장할때 사용한다,
3. 동적 상속
    - 생성자 함수를 상속받아 클래스를 확장
4. super 키워드
    - super를 호출시 수퍼클래스의 constructor를 호출한다
    - 반드시 서브클래스의 constructor에서만 호출해야한다.
    - super 호출전에 this를 참조할 수 없다.
    - 내부슬롯 HomeObject를 가진다

상속클래스의 인스턴스 생성과정

서브클래스는 자신이 직접 인스턴스를 생성하지않고 수퍼클래스에게 인스턴스 생성을 위임한다.

다만

new.target는 서브클래스를 가리킨다.

생성시 super를 호출하지않으면

1. 인스턴스가 생성되지않음
2. this 바인딩 불가

서브클래스로 인스턴스 생성시 super를 호출해야 한다